CLAIMS

What is claimed is:

1. A method for performing a transaction on a database, the method comprising:

sending a set of database modifications requested by the transaction to a first database;

placing a message in one or more message queues, said message indicating objects inserted, updated, or deleted in the transaction;

sending a commit command to the first database; and

sending said set of database modifications and a commit command to a second database.

2. The method of claim 1, further comprising:

inserting a record for the transaction into a transaction ID table in the first database.

3. The method of claim 2, wherein said sending a set of database modifications and said inserting are performed in the same transaction.

4. The method of claim 1, wherein the method is performed by an application server.

5. The method of claim 4, further comprising:

sending a cache synchronization message to other application servers sharing the same cluster as said application server.

6. The method of claim 1, wherein said set of database modifications comprises a set of structure query language (SQL) insert, update, and/or delete commands.

7. The method of claim 1, wherein said message contains a serialized representation of objects inserted, updated, or deleted in the transaction.

8. The method of claim 2, wherein said message contains a serialized representation of objects inserted, updated, or deleted in the transaction.

9. The method of claim 8, wherein said serialized representation further includes said insert of said record.

10. The method of claim 1, further comprising:

indexing messages contained in said message queue for rapid access.

11. The method of claim 5, further comprising:

receiving said cache synchronization message at another application server;

extracting a transaction ID from said cache synchronization message; and

discarding messages containing said transaction ID from one or more message queues.

12. The method of claim 2, further comprising:

periodically deleting old rows from said transaction ID table.

13. The method of claim 12, wherein said periodically deleting is performed using a background thread.

14. The method of claim 5, wherein said sending said set of database modifications and a commit command to a second database and said sending a cache synchronization message are performed asynchronously on separate threads.

15. The method of claim 5, further comprising:

detecting a failure of said first database;

halting completion of the transaction in said first database;

including in said cache synchronization message an indication that said first database is down; and

refraining from performing further actions involving said first database until said first database is restored.

16. The method of claim 15, further comprising:

replaying said database inserts, updates, and/or deletes in said cache synchronization message at a recovery server when said first database is restored.

17. The method of claim 5, further comprising:

detecting a failure of said second database;

including in said cache synchronization message an indication that said second database is down; and

refraining from performing further actions involving said second database until said second database is restored.

18. The method of claim 2, further comprising:

detecting a failure of a first recovery server;

detecting reactivation of said failed first recovery server;

reading a transaction ID out of any queued messages in a message queue corresponding to said first recovery server; and

deleting any message in said message queue that has a transaction ID matching a transaction ID in a corresponding row of said transaction ID table.

19. The method of claim 1, further comprising:

detecting a failure of a message queue;

detecting reactivation of said failed message queue;

deleting any messages in said failed message queue;

sending a message to a recovery server containing a time stamp of a first new message processed by said message queue;

receiving a message from said recovery server indicating that an oldest message still in its queue is not older than said time stamp; and

resuming normal operation upon receipt of said message from said recovery server.

20. The method of claim 1, further comprising:

detecting a failure of an application server;

determining if said failure was detected during a communication with a first database or message queue;

aborting the transaction if said failure was detected during a communication with a first database or message queue;

determining if a message has been in a message queue for a predefined period of time; and

discarding said message if a transaction ID for said message is not contained in a transaction ID table in said first database; and

replaying said set of database modifications to said second database if a transaction ID for said message is contained in said transaction ID table in said first database but not in a transaction ID table in said second database.

21. A method for failover from a failure of a first database during processing of a transaction, the transaction requesting a set of database modifications, the method comprising:

detecting a failure in the first database;

placing a message in one or more message queues, said message indicating objects inserted, updated, or deleted in the transaction, unless said message has already been placed;

sending the requested set of database modifications and a commit command to a second database; and

avoiding the first database in future transaction until the first database is restored.

22. The method of claim 21, wherein the method is performed by an application server.

23. The method of claim 22, further comprising:

sending a cache synchronization message to other application servers sharing the same cluster as said application server, said cache synchronization message including an indication that the first database is down.

24. The method of claim 21, wherein the set of database modifications comprises a set of SQL insert, update, and/or delete commands.

25. The method of claim 21, wherein said message contains a serialized representation of objects inserted, updated, or deleted in the transaction.

26. The method of claim 21, further comprising:

indexing messages contained in said message queue for rapid access.

27. The method of claim 23, further comprising:

receiving said cache synchronization message at another application server;

extracting a transaction ID from said cache synchronization message; and

discarding messages containing said transaction ID from one or more message queues.

28. The method of claim 23, wherein said sending the requested set of database modifications and a commit command to a second database and said sending a cache synchronization message are performed asynchronously on separate threads.

29. A method for failover from a failure of a second database during processing of a transaction, the transaction requesting a set of database modifications, the method comprising:

detecting a failure in the second database;

sending a cache synchronization message to other application servers sharing the same cluster as an application server executing the method, said cache synchronization message including an indication that the second database is down; and

avoiding the second database in future transaction until the first database is restored.

30. The method of claim 29, wherein the set of database modifications comprises a set of SQL insert, update, and/or delete commands.

31. The method of claim 29, further comprising:

receiving said cache synchronization message at another application server;

extracting a transaction ID from said cache synchronization message; and

discarding messages containing said transaction ID from one or more message queues.

32. A method for restoring from a failure of a first recovery server, the method comprising:

detecting reactivation of the first recovery server;

reading a transaction ID out of any queued messages in a message queue corresponding to the first recovery server; and

deleting any message in the message queue having a transaction ID matching a transaction ID in a transaction ID table in a first database.

33. A method for restoring from a failure of a message queue, the method comprising:

detecting reactivation of the failed message queue;

deleting any messages in the failed message queue;

sending a message to a recovery server containing a time stamp of a first new message processed by the message queue;

receiving a message from said recovery server indicating that an oldest message still in its queue is not older than said time stamp; and

resuming normal operation upon receipt of said message from said recovery server.

34. A method for failover from a failure of an application server during a transaction, the method comprising:

determining if a message has been in a message queue for a predefined period of time;

discarding said message if a transaction ID for said message is not contained in a transaction ID table in a first database and said message has been in a message queue for a predefined period of time; and

replaying a set of database modifications to a second database if a transaction ID for said message is contained in said transaction ID table in said first database but not in a transaction ID table in said second database.

35. The method of claim 34, further comprising:

determining if the failure was detected during a communication with a first database or message queue; and

aborting the transaction if the failure was detected during a communication with a first database or message queue.

36. An apparatus for performing a transaction on a database, the apparatus comprising:

a first database modification sender;

a message queue message inserter coupled to said first database modification sender;

a first database commit command sender coupled to said message queue message inserter; and

a second database modification and commit command sender coupled to said first database commit command sender.

37. The apparatus of claim 36, further comprising:

a database transaction ID inserter coupled to said first database modification sender and to said second database modification and commit command sender.

38. The apparatus of claim 36, wherein the apparatus is located on an application server.

39. The apparatus of claim 38, further comprising:

a cache synchronization message application server sender coupled to said second database modification and commit command sender.

40. The apparatus of claim 36, further comprising:

a message queue message indexer coupled to said message queue message inserter.

41. The apparatus of claim 36, further comprising:

a periodic transaction ID table old row deleter coupled to said first database modification sender and to said second database modification and commit command sender.

42. An apparatus for failover of a first database during processing of a transaction, the transaction requesting a set of database modifications, the apparatus comprising:

a first database failure detector;

a message queue message inserter coupled to said first database failure detector;

a second database modification and commit command sender coupled to said message queue message inserter; and

a first database avoider coupled to said first database failure detector.

43. The apparatus of claim 42, wherein the apparatus is located on an application server.

44. The apparatus of claim 42, further comprising:

a cache synchronization message application server sender coupled to said first database failure detector.

45. The apparatus of claim 42, further comprising:

a message queue message indexer coupled to said message queue message inserter.

46. An apparatus for failover from a failure of a second database during processing of a transaction, the transaction requesting a set of database modifications, the apparatus comprising:

a second database failure detector;

a cache synchronization message application server sender coupled to said second database failure detector; and

a second database avoider coupled to said second database failure detector.

47. An apparatus for restoring from a failure of a first recovery server, the apparatus comprising:

a first recovery server reactivation detector;

a message queue transaction ID reader coupled to said first recovery server reactivation detector; and

a message queue message deleter coupled to said message queue transaction ID reader.

48. An apparatus for restoring from a failure of a message queue, the apparatus comprising:

a failed message queue reactivation detector;

a failed message queue message deleter coupled to said failed message queue reactivation detector;

a recovery server time stamp message sender coupled to said failed message queue reactivation detector;

a recovery server message receiver; and

a normal operation resumer coupled to said failed message queue reactivation detector and to said recovery server message receiver.

49. An apparatus for failover from a failure of an application server during a transaction, the apparatus comprising:

an application server failure detector;

a predefined period of time message queue message determiner coupled to said application server failure detector;

a message queue message discarder coupled to said predefeined period of time message queue message determiner; and

a second database modification replayer coupled to said message queue message discarder.

50. The apparatus of claim 49, further comprising:

a communication with first database failure detector coupled to said application server failure detector and to said predefined period of time message queue message determiner; and

a transaction aborter coupled to said communication with first database failure detector.

51. An apparatus for performing a transaction on a database, the apparatus comprising:

means for sending a set of database modifications requested by the transaction to a first database;

means for placing a message in one or more message queues, said message indicating objects inserted, updated, or deleted in the transaction;

means for sending a commit command to the first database; and

means for sending said set of database modifications and a commit command to a second database.

52. The apparatus of claim 51, further comprising:

means for inserting a record for the transaction into a transaction ID table in the first database.

53. The apparatus of claim 52, wherein said sending a set of database modifications and said inserting are performed in the same transaction.

54. The apparatus of claim 51, wherein the apparatus is located on an application server.

55. The apparatus of claim 54, further comprising:

means for sending a cache synchronization message to other application servers sharing the same cluster as said application server.

56. The apparatus of claim 51, wherein said set of database modifications comprises a set of structure query language (SQL) insert, update, and/or delete commands.

57. The apparatus of claim 51, wherein said message contains a serialized representation of objects inserted, updated, or deleted in the transaction.

58. The apparatus of claim 52, wherein said message contains a serialized representation of objects inserted, updated, or deleted in the transaction.

59. The apparatus of claim 58, wherein said serialized representation further includes said insert of said record.

60. The apparatus of claim 51, further comprising:

means for indexing messages contained in said message queue for rapid access.

61. The apparatus of claim 55, further comprising:

means for receiving said cache synchronization message at another application server;

means for extracting a transaction ID from said cache synchronization message; and

means for discarding messages containing said transaction ID from one or more message queues.

62. The apparatus of claim 52, further comprising:

means for periodically deleting old rows from said transaction ID table.

63. The apparatus of claim 62, wherein said periodically deleting is performed using a background thread.

64. The apparatus of claim 55, wherein said sending said set of database modifications and a commit command to a second database and said sending a cache synchronization message are performed asynchronously on separate threads.

65. The apparatus of claim 55, further comprising:

means for detecting a failure of said first database;

means for halting completion of the transaction in said first database;

means for including in said cache synchronization message an indication that said first database is down; and

means for refraining from performing further actions involving said first database until said first database is restored.

66. The apparatus of claim 65, further comprising:

means for replaying said database inserts, updates, and/or deletes in said cache synchronization message at a recovery server when said first database is restored.

67. The apparatus of claim 55, further comprising:

means for detecting a failure of said second database;

means for including in said cache synchronization message an indication that said second database is down; and

means for refraining from performing further actions involving said second database until said second database is restored.

68. The apparatus of claim 62, further comprising:

means for detecting a failure of a first recovery server;

means for detecting reactivation of said failed first recovery server;

means for reading a transaction ID out of any queued messages in a message queue corresponding to said first recovery server; and

means for deleting any message in said message queue that has a transaction ID matching a transaction ID in a corresponding row of said transaction ID table.

69. The apparatus of claim 51, further comprising:

means for detecting a failure of a message queue;

means for detecting reactivation of said failed message queue;

means for deleting any messages in said failed message queue;

means for sending a message to a recovery server containing a time stamp of a first new message processed by said message queue;

means for receiving a message from said recovery server indicating that an oldest message still in its queue is not older than said time stamp; and

means for resuming normal operation upon receipt of said message from said recovery server.

70. The apparatus of claim 51, further comprising:

means for detecting a failure of an application server;

means for determining if said failure was detected during a communication with a first database or message queue;

means for aborting the transaction if said failure was detected during a communication with a first database or message queue;

means for determining if a message has been in a message queue for a predefined period of time; and

means for discarding said message if a transaction ID for said message is not contained in a transaction ID table in said first database; and

means for replaying said set of database modifications to said second database if a transaction ID for said message is contained in said transaction ID table in said first database but not in a transaction ID table in said second database.

71. A apparatus for failover from a failure of a first database during processing of a transaction, the transaction requesting a set of database modifications, the apparatus comprising:

means for detecting a failure in the first database;

means for placing a message in one or more message queues, said message indicating objects inserted, updated, or deleted in the transaction, unless said message has already been placed;

means for sending the requested set of database modifications and a commit command to a second database; and

means for avoiding the first database in future transaction until the first database is restored.

72. The apparatus of claim 71, wherein the apparatus is located on an application server.

73. The apparatus of claim 72, further comprising:

means for sending a cache synchronization message to other application servers sharing the same cluster as said application server, said cache synchronization message including an indication that the first database is down.

74. The apparatus of claim 71, wherein the set of database modifications comprises a set of SQL insert, update, and/or delete commands.

75. The apparatus of claim 71, wherein said message contains a serialized representation of objects inserted, updated, or deleted in the transaction.

76. The apparatus of claim 71, further comprising:

means for indexing messages contained in said message queue for rapid access.

77. The apparatus of claim 73, further comprising:

means for receiving said cache synchronization message at another application server;

means for extracting a transaction ID from said cache synchronization message; and

means for discarding messages containing said transaction ID from one or more message queues.

78. The apparatus of claim 73, wherein said sending the requested set of database modifications and a commit command to a second database and said sending a cache synchronization message are performed asynchronously on separate threads.

79. An apparatus for failover from a failure of a second database during processing of a transaction, the transaction requesting a set of database modifications, the apparatus comprising:

means for detecting a failure in the second database;

means for sending a cache synchronization message to other application servers sharing the same cluster as an application server executing the method, said cache synchronization message including an indication that the second database is down; and

means for avoiding the second database in future transaction until the first database is restored.

80. The apparatus of claim 79, wherein the set of database modifications comprises a set of SQL insert, update, and/or delete commands.

81. The apparatus of claim 79, further comprising:

means for receiving said cache synchronization message at another application server;

means for extracting a transaction ID from said cache synchronization message; and

means for discarding messages containing said transaction ID from one or more message queues.

82. An apparatus for restoring from a failure of a first recovery server, the apparatus comprising:

means for detecting reactivation of the first recovery server;

means for reading a transaction ID out of any queued messages in a message queue corresponding to the first recovery server; and

means for deleting any message in the message queue having a transaction ID matching a transaction ID in a transaction ID table in a first database.

83. An apparatus for restoring from a failure of a message queue, the apparatus comprising:

means for detecting reactivation of the failed message queue;

means for deleting any messages in the failed message queue;

means for sending a message to a recovery server containing a time stamp of a first new message processed by the message queue;

means for receiving a message from said recovery server indicating that an oldest message still in its queue is not older than said time stamp; and

means for resuming normal operation upon receipt of said message from said recovery server.

84. An apparatus for failover from a failure of an application server during a transaction, the apparatus comprising:

means for determining if a message has been in a message queue for a predefined period of time;

means for discarding said message if a transaction ID for said message is not contained in a transaction ID table in a first database and said message has been in a message queue for a predefined period of time; and

means for replaying a set of database modifications to a second database if a transaction ID for said message is contained in said transaction ID table in said first database but not in a transaction ID table in said second database.

85. The apparatus of claim 84, further comprising:

means for determining if the failure was detected during a communication with a first database or message queue; and

means for aborting the transaction if the failure was detected during a communication with a first database or message queue.

86. A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform a method for performing a transaction on a database, the method comprising:

sending a set of database modifications requested by the transaction to a first database;

placing a message in one or more message queues, said message indicating objects inserted, updated, or deleted in the transaction;

sending a commit command to the first database; and

sending said set of database modifications and a commit command to a second database.

87. A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform a method for failover from a failure of a first database during processing of a transaction, the transaction requesting a set of database modifications, the method comprising:

detecting a failure in the first database;

placing a message in one or more message queues, said message indicating objects inserted, updated, or deleted in the transaction, unless said message has already been placed;

sending the requested set of database modifications and a commit command to a second database; and

avoiding the first database in future transaction until the first database is restored.

88. A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform a method for failover from a failure of a second database during processing of a transaction, the transaction requesting a set of database modifications, the method comprising:

detecting a failure in the second database;

sending a cache synchronization message to other application servers sharing the same cluster as an application server executing the method, said cache synchronization message including an indication that the second database is down; and

avoiding the second database in future transaction until the first database is restored.

89. A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform a method for restoring from a failure of a first recovery server, the method comprising:

detecting reactivation of the first recovery server;

reading a transaction ID out of any queued messages in a message queue corresponding to the first recovery server; and

deleting any message in the message queue having a transaction ID matching a transaction ID in a transaction ID table in a first database.

90. A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform a method for restoring from a failure of a message queue, the method comprising:

detecting reactivation of the failed message queue;

deleting any messages in the failed message queue;

sending a message to a recovery server containing a time stamp of a first new message processed by the message queue;

receiving a message from said recovery server indicating that an oldest message still in its queue is not older than said time stamp; and

resuming normal operation upon receipt of said message from said recovery server.

91. A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform a method for failover from a failure of an application server during a transaction, the method comprising:

determining if a message has been in a message queue for a predefined period of time;

discarding said message if a transaction ID for said message is not contained in a transaction ID table in a first database and said message has been in a message queue for a predefined period of time; and

replaying a set of database modifications to a second database if a transaction ID for said message is contained in said transaction ID table in said first database but not in a transaction ID table in said second database.